Kontrolltöö tulemused

Punktide vahemik	Tudengite arv
$[0, 10)$	2
$[10, 15)$	8
$[15, 20)$	13
$[20, 25)$	17
$[25, 27]$	33

Ülesanne

Kuidas implementeerida take?

```
take : Nat → Stream a → List a
take n xs = ?impl
```

Näited:

```
Main> take 5 [1..]
[1,2,3,4,5]
Main> take 5 [1,3..]
[1, 3, 5, 7, 9]
```

Ülesanne

Kuidas implementeerida tingimusavaldis funktsiooniga cond?

```
cond : Bool → a → a → a
cond True  x y = x
cond False x y = y
```

Mis probleemid tekivad antud definitsiooniga?

Lõpmatud andmetüübid

- Idris võimaldab defineerida lõpmatuid (vahe) struktuure.
- NB! Lõpmatuid struktuure ei saa täielikult välja arvutada.
- Täieliku arvutamise piiramiseks peab struktuuri lõpmatu osa olema `Inf`-tüübi taga. Vt. `Stream`
- Idris teisendab ise: `Inf a` $\leftrightarrow$ `a`

- Striimid (`Stream`) lõpmatud, "laisad" listid. Näiteks

```
arvudAlates : Int → Stream Int
arvudAlates n = n :: arvudAlates (n+1)
```

- Striimide ainus konstruktor on `(::) : a → Inf (Stream a) → Stream a`
- Meenuta, et listidel on lisaks tühja listi konstruktor `[] : List a`.

- Funktsioon `take n s` tagastab lõpmatu striimi `s` esimesed n elementi. Näiteks: `take 5 (arvudAlates 7)` on `[7, 8, 9, 10, 11]`

Enumeratioonid

- `[m..]` on striim $[m, m+1, m+2, \ldots]$

  Näiteks: `take 10 [2..]` on `[2, 3, 4, 5, 6, 7, 8, 9, 10, 11]`

- `[m,n..]` on striim $[m, m+(n-m), m+2(n-m), \ldots]$

  Näiteks: `take 10 [2,5..]` on `[3, 5, 7, 9, 11, 13, 15, 17, 19, 21]`

Laisad tüübid

Sarnaselt `Inf` tüüpidele saab optimeerimiseks kasutada `Lazy` tüüpe:

```
data Lazy : Type → Type where
     Delay : a → Lazy a
```

... ja funktsioon

```
Force : Lazy a → a
```

Reeglid:

- `Delay` argumenti ei väärtustata.
- `Force (Delay e) = e`
- Idris teisendab väärtusi vastaval tüübile `Lazy a ↔ a`

Soovitus: Pane `Lazy` argumendi tüübi ümber, kui seda ei pruugi vaja minna.
Näiteks: `(&&) : Bool → Lazy Bool → Bool`.